Vesnina Anastasia Alexandrovna
Ural Federal University
Russia, Yekaterinburg
Research advisor: Kovaleva Alexandra Georgievna

AN APPROACH TO UNIFICATION OF APPLICATION PROGRAMMING INTERFACES OF GAMING PLATFORMS FOR ARTIFICIAL INTELLIGENCE

Abstract: *This paper explores several existing gaming platforms for training and testing artificial intelligence. Application programming interfaces of these platforms are analyzed in order to discover potential complications of porting intellectual systems between them. An approach to unification of programming interfaces in order to overcome these complications is presented. Advantages and potential side effects of this approach are described.*

Keywords: *artificial intelligence, machine learning, deep learning, reinforcement learning, video games, application programming interface.*

Веснина Анастасия Александровна
Уральский федеральный университет
Россия, г. Екатеринбург
Научный руководитель: Ковалева Александра Георгиевна

ПОДХОД К УНИФИКАЦИИ ПРОГРАММНЫХ ИНТЕРФЕЙСОВ ИГРОВЫХ ПЛАТФОРМ ДЛЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Аннотация: *Статья обозревает существующие игровые платформы для обучения и тестирования искусственного интеллекта. Анализируются программные интерфейсы этих платформ для того, чтобы раскрыть возможные осложнения при портировании интеллектуальных систем между*

ними. Статья представляет подход к унификации программных интерфейсов, который позволит преодолеть эти осложнения. В статье представлены преимущества и потенциальные побочные эффекты этого подхода.

Ключевые слова: *искусственный интеллект, машинное обучение, глубокое обучение, обучение с закреплением, видеоигры, программный интерфейс***.**

Introduction

Artificial intelligence (AI) is a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation, including tasks that are very hard for machine, like pattern recognition [1]. Machine learning (ML) is a subset of artificial intelligence that studies algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions [2]. Games are an interesting area of application of AI, and deep learning technologies are currently being adopted in them. It is not uncommon to use game environment to train and test AI that would be hard or even dangerous to test in real environment that is modelled. For example, GTA video game series have been used for solving computer vision tasks and training car-driving AI in city environment [3].

At the same time, video games are rarely optimized for machine learning tasks because of the limited runtime speed and lack of the suitable application programming interface (API) for other computer programs.

Existing platforms overview

Several gaming platforms are chosen in order to bring attention to complications caused by their features that may rise when trying to port intellectual system (IS) trained on one platform to another.

OpenAI [4] currently maintains wrapper for video game emulator cores of several retro game consoles, thus providing thousands of games for these consoles, adapted for the needs of game AI researchers and developers. However, API of these games mainly consists of collections of variables, their types and memory addresses, which results in couple of complications. Firstly, convenient game state presentation for AI algorithms from such data set may become additional complex task for

developers. Secondly, such data presentation is not suitable for modern video games, because they are developed using different technologies that cannot guarantee accurate memory addressing for variables, which will not allow to generalize developed software.

DeepRTS is a platform for research, training and testing of AI algorithms for games in real-time strategy genre (RTS) based on deep learning and reinforcement learning techniques [5]. The feature of this platform is headless mode without graphical interface. According to the tests results (Table 1) [5], this mode enables game to run at the speed of several millions FPS, several orders of the magnitude higher than usual. This allows developers to speed up greatly the process machine learning system training. The platform supplies game state data through API in form of three-dimensional matrix, where each layer is a set of information of specific kind for each point in the game space.

Table 1. Runtime speed comparison of different platforms

Platform	FPS
OpenAI Gym	60
SC2LE	60-144
DeepRTS	24,000-7,000,000

Developers that are seriously interested in AI for RTS video games may be interested in API presented by Blizzard Entertainment for video game StarCraft II, which is considered to be RTS etalon nowadays, and reinforcement learning environment around this API developed by Deepmind Ltd [6]. StarCraft II is really hard trial for AI developers due to its game mechanics complexity, the number of possible state variables and reward delay. At current time there is no possibility to run game in headless or software rendering mode, which creates bottleneck in data transfer between CPU and GPU and limits runtime speed and thus algorithm training speed.

This may result in additional difficulty in algorithm training. StarCraft II API is presented using Protocol Buffer technology, which gives a wide choice of tools for developing software.

Possible difficulties

DeepRTS and StarCraft II examples make us consider various facts. These video games are very similar at the core: they are both classic exemplars of RTS with lots of similarities in game mechanics, game state presentation for player, etc. At the same time, if IS developers for DeepRTS decide to port it to StarCraft II and retrain it for new platform, they would have to develop a layer between their system and gaming platform from scratch. They may not even have one: they may have not predicted they would ever port it to another platform and they could train their model on specific data presentation that is supplied by DeepRTS developers.

Additionally, if StarCraft II developers would ever be interested in using the IS developed for DeepRTS as their game component, they would possibly have to either change this system's API or their own inner logic of interaction between game system and AI system.

Porting AI from DeepRTS to StarCraft II platform may be additionally complicated by harsh difference of system requirements between platforms: lack of headless mode in StarCraft II would increase hardware load greatly and decrease algorithm training speed.

Approach for overcoming difficulties

The possible solution for overcoming these difficulties is in creating an approach that would unify APIs of platforms on which IS are trained and tested, and of IS, which can be integrated into these platforms.

It would be extremely hard to develop single approach for all platforms, of course, because video games may differ so much they would not be able to present similar API effectively. Game state presentation of chess and racing game would differ as much as they differ themselves. It would be more practical to develop video games classification by game state form and game process, which would allow us to generalize platform by a set of parameters suitable for particular genre and video game.

Table 2. Examples of methods provided by game platform

Method name	Platform parameter defining this method
subscribeToStateUpdate	Turn-based game process
applyMovementVector	Personalized controls, continuous game space
getVisibleObjects	Personalized game state presentation

The following examples of criterions may be in this classification:

- Whether game process is turn-based (e. g. chess) or real-time (StarCraft series);
- Whether game space is sampled (chess) or nearly continuous (GTA series);
- Whether game state presentation is personalized, that is, it features first-person of third-person view (GTA series), or generalized, for example with full overview of game space or with top-down view (chess, StarCraft series).

Games falling in different categories obviously require different approaches to development and training of IS for them. Some examples of methods provided by APIs of games falling in different categories are presented in Table 2.

This approach may pose some minimal but principle requirements general to all gaming platforms, independent of genre or other criterions. Examples of such requirements include presence of headless mode without GPU usage and general format definition for game state presentation. Some uniform API description form and method calling protocol may also become common, such as interface definition language. Existing one may even be used, such as Apache Thrift [7].

That way, gaming platform developers may be able to classify their project by several criterions and find an approach to provide API in the most uniform way. This would allow simplifying process of porting IS from other platform which has the same parameters in these criterions.

Conclusion

The developed approaches may open the possibility for growth of game AI development without coupling to specific game environment with possible adaptation by contract, which would decrease development costs.

In case of using game environment for the AI training for real world problems, ability to use several different game environments may help AI developers. Suggested approach is intended to simplify the process of transition between systems, giving opportunity for the use of several different environments.

REFERENCES

1. Kaplan, M. Haenlein, Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence, Business Horizons Volume 62, Issue 1, January–February 2019, Pages 15-25
2. Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2
3. S. R. Richter, V. Vineet, S. Roth, V. Koltun, Playing for Data: Ground Truth from Computer Games, European Conference on Computer Vision, October 8-16, 2016, Amsterdam, pp. 102-118.
4. OpenAI Gym Retro. https://github.com/openai/retro
5. P. Andersen, M. Goodwin and O. Granmo, «Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games», 2018 IEEE Conference on Computational Intelligence and Games (CIG), Maastricht, 2018, pp. 1-8.
6. O. Vinyals T. Ewalds S. Bartunov P. Georgiev A. S. Vezhnevets M. Yeo A. Makhzani H. Küttler J. Agapiou J. Schrittwieser et al. «StarCraft II: a new challenge for reinforcement learning» arXiv preprint arXiv:1708.04782 2017.
7. Apache Thrift (2017). Apache Thrift documentation. https://thrift.apache.org/docs